club commodore

Boletín informativo para los usuarios de microordenadores **VIC y CBM**



LA NUEVA IMPRESORA MODELO VIC-1525



Nº/13 noviembre 1983

MAPAS DE MEMORIA DEL COMMODORE 64

E000-FFFF	8K KERNAL ROM OR RAM
D000-DFFF	4K I/O OR RAM OR CHARACTER ROM
C000-CFFF	4K RAM
A000-BFFF	8K BASIC ROM OR RAM OR ROM PLUG-IN
8000-9FFF	8K RAM OR ROM PLUG-IN
4000-7FFF	16K RAM
0000-3FFF	16K RAM

I/O BREAKDOWN

D000-D3FF	VIC (Video Controller)	1K Bytes
D400-D7FF	SID (Sound Synthesizer)	1K Bytes
D800-DBFF	Color RAM	1K Nybbles
DC00-DCFF	CIA1 (Keyboard)	256 Bytes
DD00-DDFF	CIA2 (Serial Bus, User Port/RS-232)	256 Bytes
DE00-DEFF	Open I/O slot #1 (CP/M Enable)	256 Bytes
DF00-DFFF	Open I/O slot #2 (Disk)	256 Bytes

E000	8K KERNAL ROM
D000	4K I/O
C000	4K RAM (BUFFER)
A000	8K BASIC ROM
8000	8K RAM
4000	16K RAM
0000	16K RAM

X = DON'T CARE
0 = LOW
1 = HIGH

LORAM = 1
HIRAM = 1
GAME = 1
EXROM = 1

This is the default BASIC memory map which provides BASIC 2.0 and 38K contiguous bytes of user RAM.

E000	8K RAM
D000	4K I/O
C000	4K RAM
8000	16K RAM
4000	16K RAM
0000	16K RAM

X = DON'T CARE
0 = LOW
1 = HIGH

LORAM = 1
HIRAM = 0
GAME = 1
EXROM = X

OR

LORAM = 1
HIRAM = 0
GAME = 0

(THE CHARACTER ROM IS NOT ACCESSIBLE BY THE CPU IN THIS MAP)

EXROM = 0

This map provides 60K bytes of RAM and I/O devices. The user must write his own I/O driver routines.

Depósito Legal Sep. B. 2133-1958 - Altés, s.l., Caballero 87, Barcelona

EDITORIAL

la paradoja del programador

Generalmente —en microinformática— solemos preocuparnos de un montón de problemas relacionados con los lenguajes, tamaño y utilización de la memoria, trucos para escribir mejores programas, etc... Pero, sólo de tarde en tarde, nos paramos a pensar en el papel que estamos desarrollando al darle —con mayor o menor fruición— a las teclas de nuestro microordenador.

Un buen tema de meditación es la llamada "paradoja del programador" que puede servirnos para aclarar ciertos conceptos —que muy posiblemente nuestros lectores tengan claros, aunque mucha gente "de fuera" no— que son fundamentales a la hora de saber "a qué estamos jugando". Esta paradoja consiste en lo siguiente: se entiende comúnmente que programamos a los ordenadores para que nos resuelvan problemas. Ahora bien, también es de dominio público que no suele darse el caso de que un programa funcione perfectamente a la primera y que aún cuando hayamos depurado los errores que detecta el propio lenguaje (SYNTAX ERRORs y demás fastidiosas circunstancias), nadie nos garantiza que alguna de las fórmulas que hemos introducido no tenga un signo cambiado o no esté mal planteada de buen principio, de manera que es necesario establecer algún sistema para verificar nuestro programa que pasa —necesariamente— por un buen conocimiento del problema a resolver. Lo que nos lleva a la conclusión siguiente: para programar un ordenador, para resolver un problema, es necesario que sepamos resolver este problema previamente, lo que, a su vez, nos obliga a preguntarnos ¿para qué demonios necesitamos un ordenador para resolver problemas si necesitamos saber resolverlos previamente?

La respuesta a esta pregunta aclara el papel del ordenador en nuestro trabajo o nuestro entretenimiento pues nos brinda la oportunidad de corregir un error muy frecuente que suele cometer la gente cuando habla de ordenadores: los ordenadores no resuelven problemas POR SÍ SOLOS, simplemente se limitan a repetir de forma rápida una serie de instrucciones que ha escrito en su memoria previamente un programador.

Por lo que es falso que los ordenadores resuelvan problemas, son simplemente "extensiones" o "periféricos" de la inteligencia del programador (¡OJO! cuando hablamos del "programador" —en singular— nos estamos refiriendo a una entidad de una o muchas personas, trabajando individualmente o en equipo, dependiendo de la complejidad de la tarea a resolver). De esta manera, alguien ha llegado a hablar de los ordenadores como "amplificadores de inteligencia" lo cual, dada la innata estupidez de estas máquinas, no deja de ser curioso, siendo más ajustada a la realidad una definición de los ordenadores —algo cínica— que dice que son "idiotas rápidos".

En resumen: quien programa el ordenador debe conocer a fondo el problema que va a tratar o debe estar en contacto con alguien que posea este conocimiento, debe escribir su programa de forma que pueda verificar que todas las respuestas de la máquina a variables conocidas son correctas y, que por extensión, las respuestas a circunstancias desconocidas serán correctas. También debe asegurarse que su programa —como mínimo— no responderá a circunstancias imprevistas o imposibles y, para terminar, esta persona o grupo deberá poseer un buen conocimiento de la herramienta que va a utilizar, teniendo muy claras sus limitaciones.

Para finalizar, una propuesta: si alguien está interesado en abrir un debate sobre estos u otros aspectos de la utilización de la microinformática, que no dude en hacernos llegar sus opiniones. En CLUB COMMODORE creemos que hace falta un intercambio de puntos de vista por parte de los propios usuarios, y —en la medida que podamos— queremos contribuir a la averiguación de "qué pito vamos a tocar". ■

VENTANA CBM

READY (del chip a la base de datos)

por **RAFAEL NAVARRO** (M.E.C. SOFT.)

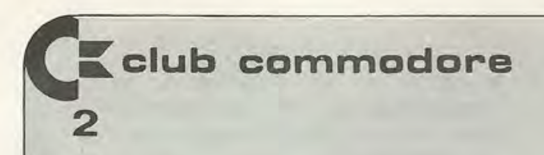
No hay, en un ordenador, actitud más persistente que la que adopta cuando, a la espera de nuestras órdenes, nos informa de que se encuentra preparado (READY), poniendo su cursor a nuestra disposición.

Otra actitud muy frecuente es la de presentarnos una serie de opciones de trabajo. Fácilmente, el usuario puede ordenar la consecución de cualquiera de ellas mediante una sencilla operación (pulsación de una tecla, activación de un interruptor, etc.)

En otras ocasiones, las cosas son mucho más intuitivas: uno se encuentra azarosamente introducido en un vertiginoso conflicto intergaláctico cuyas condicionantes de tiempo y agresividad no dan lugar a la vacilación. Cualquier fallo acabará con nuestras naves, nuestras vidas y, muy probablemente, con nuestra reputación como aguerridos luchadores del espacio.

Existe, finalmente, una última visión popular de la informática: el ordenador genio-resuelvelo todo que

(continúa en la pág. siguiente)



VENTANA CBM

READY (del chip a la base de datos)

(continuación)

aparece en los documentales científicos de la televisión y algunas revistas. La idea que se nos da de los sistemas es la de un producto final, nacido perfecto e infalible. Nada más lejos de la realidad. Tras la gruesa capa de maquillaje y los mil emperifollamientos de ergonomía, color y diseño se hallan los seres más tontos de la creación.

Estas y otras visiones pertenecen a niveles distintos de evolución. Ésta es la idea básica que pretendo desarrollar en esta serie de artículos. A pesar de lo dicho, es absolutamente cierto que la informática resuelve para el hombre problemas de importancia, ya sea por su complejidad o por lo rutinario de los mismos. Ahora bien, para poder realizar estos trabajos, las máquinas informáticas, el llamado hardware, requiere un concienzudo entrenamiento. Como en un inteligente perro adiestrado para servir al hombre, las secuencias de instrucciones son almacenadas en los ordenadores para ser ejecutadas cuando se requiera.

Esta serie tratará precisamente de eso; de los niveles y tipos de conocimientos que es necesario introducir en un ordenador para que nos resulte útil. Me referiré, salvo excepciones ilustrativas, al mundo de los microordenadores que es, por el momento, el área de producción de COMMODORE, entendiendo por tales a los sistemas basados en uno o varios microprocesadores.

I. LA ENSEÑANZA PRIMARIA

De nada servirían las capacidades fundamentales de decisión y cálculo (SI/NO) de un microprocesador sin un rígido control basado en una serie de instrucciones elementales que el mismo sea capaz de asimilar y comprender. De este modo, todo fabricante de microprocesadores especifica, junto a las características electrónicas, las órdenes que sus productos son capaces de entender y ejecutar, al igual que los soportes internos de las mismas (los registros). Así, desde el punto de vista del software (capacidad de ser programado), un micro, consta de las siguientes partes fundamentales (recordemos que estamos al nivel más bajo):

a) unidad de control o secuenciadora;
b) unidad aritmético-lógica (álgebra de Boole);
c) decodificador de instrucciones;
d) registro de datos e instrucciones;
e) registro de direcciones;
f) contador de programa;
g) acumulador y registros complementarios.

Como en cualquier orquesta sinfónica, si todos los elementos tocasen a su aire, sin el control de un director, el resultado, en un microprocesador, sería desastroso. El «director» que controla los momentos en que cada cosa debe hacerse dentro del micro es el **secuenciador** o unidad de control. El **Contador de programa** apunta a la dirección de memoria (ROM o RAM externas) que contiene la instrucción a ejecutar. Dicha dirección es leída en el **registro de direcciones,** y su contenido (la instrucción con sus operandos si los requiere) es leída en el **registro de datos e instrucciones.** De ahí, siempre con el permiso del secuenciador, pasará al **decodificador de instrucciones,** quien se encargará de ejecutar toda la serie de **microinstrucciones** previstas por el fabricante, que son necesarias para ejecutar la instrucción hallada por el **contador de programa** en la memoria donde el mismo residía. Cuando me refiero a instrucciones, en este caso, quiero decir instrucciones de **código máquina,** o código de ceros y unos, único que es capaz de digerir el microprocesador. Estas instrucciones constituyen el programa, o parte del programa, introducido por el programador. Las **microinstrucciones** son las operaciones más elementales que es capaz de ejecutar el micro, las cuales vienen dictadas por el hardware, son inalterables, y constituyen el primer soporte del programador, al permitirle trabajar en código máquina. El mecanismo de las microinstrucciones está contenido en el microprocesador y es gobernado por el decodificador de instrucciones.

Éste es, por supuesto, un nivel de trabajo considerablemente arduo que sólo es empleado en empresas de desarrollo, procesos industriales y casos en que los condicionantes de tiempo y economía son fundamentales. O, simplemente, cuando se está «empezando» a crear un nuevo sistema COMMODORE. Así pues, un microprocesador, incluso con su memoria RAM, su teclado, sus discos y su pantalla, no sirve para nada si no existe modo de introducirle algunas instrucciones y ordenarle que las ejecute.

En generaciones anteriores, éste, el nivel más bajo de programación o enseñanza, estaba soportado por las llamadas «consolas» de programación. Estos eran unos dispositivos hardware, dotados de una serie de interruptores que permitían:

a) posicionarse en un punto de la memoria;
b) visualizar y/o modificar su contenido;
c) incrementar o decrementar dicha posición;
d) ordenar la ejecución del código introducido, a partir de una posición de memoria.

De este modo podían introducirse y ejecutarse los programas o funciones básicas de un modo experimental («debbuging») hasta darlos por definitivos. Con este sistema o sistemas igualmente elementales se graban las ROMs de inicialización que contienen los «bootstraps» (programas residentes en memoria no volátil que permiten la carga de programas residentes en disquetes u otros soportes). La técnica de «bootstrap» o ROMs de inicialización, se emplea en máquinas cuyo sistema operativo (este concepto se discutirá más adelante) debe ser cargado desde fuera (no residente). En los sistemas COMMODORE, el sistema de inicialización reside, junto al sistema operativo, el interpretador BASIC y el KERNAL (input/output) en memorias no volátiles.

El sistema operativo lo constituyen una serie de programas y rutinas que permiten trabajar con el ordenador, recibir datos del teclado, imprimirlos en la pantalla u otro periférico, controlar la ejecución de los programas, etc. Ésta es, desde luego, una definición muy general, pero debe tenerse en cuenta que en ella se mezclan los conceptos semántico e informático y que, dada la rápida evolución de la microinformática, no encuentro manera de especificar más datos acerca del concepto.

el lenguaje FORTH (I)

por **RAFAEL PARDO**

En este número nuestro colaborador RAFAEL PARDO inicia una serie de artículos dedicados a un nuevo lenguaje llamado FORTH. Rafael es ya un viejo conocido de los lectores de nuestra Revista por su colaboración en la sección de Electrónica. Ahora se nos ha convertido en entusiasta del lenguaje FORTH y no ha parado hasta obtener un "hueco" en la Revista para esta nueva sección. Esperamos que nuestros lectores aficionados a las emociones fuertes apreciarán debidamente este nuevo lenguaje.

¿Qué es FORTH? Es un lenguaje creado en la década de los sesenta por Charles H. Moore, y se caracteriza por su rapidez, por ser muy compacto (un sistema típico FORTH está soportado en una ROM de 8K), y por su facilidad de bajar al nivel de lenguaje máquina.

En FORTH hay varios conceptos claves: el de diccionario, el de palabra, el «stack» o pila y la utilización del sistema de notación RPN (Reverse Polish Notation = Notación Polaca Inversa) idéntico al utilizado por las calculadoras científicas de la conocida firma Hewlett-Packard.

EL DICCIONARIO

El diccionario en FORTH, como en cualquier otro diccionario, consiste en una lista de palabras y sus definiciones, estando éstas en términos de otras palabras.

Una palabra, en FORTH, es una cadena de caracteres acabada en un espacio en blanco. FORTH no es como los otros lenguajes de programación, no tiene palabras reservadas, propias del lenguaje. Cualquier combinación de caracteres es una palabra válida. Éstas son palabras FORTH:

EDITOR . DUP EMPTY-BUFFERS FLUSH ROT + —

(continúa en la pág. siguiente)

READY (del chip a la base de datos) (conclusión)

Una vez se dispone de las funciones más elementales de un sistema operativo:

a) control de entrada/salida (pantalla, teclado, periférico);

b) monitor de código máquina

ya pueden escribirse programas en código máquina. El monitor de código máquina es un programa, siempre presente de una manera u otra en los sistemas operativos, que permite efectuar las mismas operaciones que una consola pero a través del teclado y de una manera mucho más cómoda. Considérese, en todo momento, que estamos explicando las cosas desde el principio y, en consecuencia, presuponemos que en tales circunstancias no se dispone de ninguna ayuda de software, de modo que hay que hacérselo todo.

Gracias al monitor de código máquina, podremos escribir lo que se llama un ensamblador.

EL ENSAMBLADOR

Un ensamblador es un programa que permite emplear instrucciones simbólicas en vez de meros valores numéricos para introducir instrucciones y operandos en el ordenador. De este modo, se trabaja mucho más cómodamente. Por ejemplo, escribiendo en código máquina, si deseamos escribir un programa que lea el contenido de la posición de memoria $2000 ($2000 hexadecimal = 8192 en decimal), lo cargue en el acumulador y lo escriba en la posición $2001, dicho programa tendrá que escribirse así:

FD operación de carga en el acumulador;
00 dirección de memoria u operando;
20 expresado en forma inversa;
8D operación de escritura;
01 dirección de memoria destino.
20

O, escrito en una línea, ad 00 20 8d 01 20, que es como se vería en la pantalla con el monitor.

El mismo programa adquiere mucho más significado escrito en lenguaje ensamblador:

LDA 0020 instrucción de carga del acumulador;
STA 0201 instrucción de escritura;
LDA = LoaD Accumulator — STA = STore Accumulator.

El sistema de ensamblado está compuesto por tres módulos fundamentales:

a) **editor** que permite escribir y corregir el programa;

b) **ensamblador** propiamente dicho que convierte las instrucciones de ensamblador en instrucciones de código máquina (único que entiende el ordenador) y las registra en disco, cassette, etc.

c) **loader,** que lee las instrucciones de código máquina del disco o cassette y las coloca en la memoria para ser ejecutadas y/o corregidas con el monitor.

El ensamblador nos permite ya crear una serie de programas de utilidad más directa y más cercanos a la actividad del hombre con los ordenadores:

a) programas industriales en ROM;
b) sistemas operativos sofisticados;
c) interpretadores;
d) compiladores;
e) utilitarios;
f) programas para usuario final;
g) bases de datos, etc.

De los cuales hablaremos en siguientes artículos.

el lenguaje FORTH

(continuación)

Para FORTH (*) las letras de una palabra pueden ser mayúsculas o minúsculas, cuando procesa una palabra. Así esta palabra es la misma escrita de todas estas maneras:

FLUSH fLuSh FIUsh flush

Esto permite que no tenga de pulsar SHIFT para entrar una palabra FORTH. Dentro de cadenas de caracteres o mensajes (entre comillas), FORTH reconoce las mayúsculas y las minúsculas.

El diccionario es una lista «entrelazada» de palabras compiladas. La definición de la palabra tiene una longitud variable y su contenido depende del tipo de palabra definida que queramos usar para definir la nueva palabra. El diccionario se puede extender y ocupar toda la memoria.

El diccionario se puede extender utilizando la definición de palabras. La definición más común es con los «dos puntos (:)». La ejecución de (:) hará una entrada de la palabra que siga en el diccionario. Las direcciones de todas las palabras ulteriores se compilarán en el diccionario. Se finaliza la definición cuando se encuentra un «punto y coma (;)».

Éste es un ejemplo sencillo:

: VEN 100 * . ; (return) OK
1234 VEN (return) 123400 OK

En las dos líneas del ejemplo anterior hemos realizado lo siguiente: en la primera hemos definido la palabra VEN para que nos realice de forma automática una multiplicación por 100 y nos imprima el resultado. En la segunda, hacemos uso de esta palabra que acabamos de definir, siendo 1234 el número que queremos multiplicar y —naturalmente— 123400 el resultado. La palabra VEN puede usarse como equivalente de la secuencia de palabras que la siguen, suponiendo que hayan sido previamente definidas. Si no ha conseguido captar completamente el funcionamiento de la definición de palabras en FORTH no se preocupe, la anterior explicación es sólo un ejemplo que será desarrollado extensamente en próximos artículos.

«STACK» (O PILA)

FORTH tiene dos «stacks», que se denominan **«stack» de parámetros** y **«stack» de retorno de subrutinas («Return Stack»)**. Como el «stack» de parámetros se usa mucho más que el «stack» de retorno de subrutinas, cuando nos refiramos «al stack» se deberá entender el «stack» de parámetros.

Ambos «stacks» colocan los nuevos elementos en su parte baja. El «stack» también se llama **LIFO** (Last Input First Output = Última Entrada Primera Salida).

Introduzca esto en su ordenador:

5 6 7 8 9 (return) ok

FORTH siempre responderá con **ok** cuando todos los comandos antes de (return) se han ejecutado correctamente.

Ha colocado cinco valores en el «stack». El primer valor (5) está en la parte baja del «stack», y el último valor (9) en la parte alta. Entre cinco puntos (.):

. . . . . (return)

Esto hará que FORTH responda (en la misma línea):

9 8 7 6 5 ok

La palabra punto (.) es una palabra FORTH que imprime el número más alto en la pila (el «stack») en el periférico de salida seleccionado y sustituye este valor por el inmediatamente inferior. El primer punto imprimirá el (9) y moverá hacia arriba el (8). Los siguientes puntos imprimirán los valores restantes de la misma manera y vaciarán la pila («stack»).

(continuará)

* *Siempre hablamos del «FORTH V1.0 de Datatronic AB para Commodore 64», aunque todo lo dicho puede funcionar en cualquier adaptación FORTH existente en el mercado, siempre y cuando sea una versión de la definición FIG-FORTH.*

La gracia de los cuestionarios (a mi entender), radica en que cada vez te pregunten cosas diferentes (aunque sean del mismo tema). Un cuestionario fijo acabas por aprendértelo de memoria sin que te hayas fijado en lo que dice. Así pues, para entrar en un tema cualquiera a través de un cuestionario, quizá la mejor forma sea la del azar. El azar te hace proposiciones distintas cada vez y con ellas el cuestionario te parece nuevo. También es importante el modo con el cual se formulan las preguntas. Un modo muy simple como el de proponerte varias respuestas numeradas y pedirte sólo un número de respuesta, es muy rápido de manipulación pero también muy aburrido y poco didáctico si abusas de él.

En esta edición os voy a proponer un programa relativamente simple, que hará (ésta es mi intención), que podáis imaginar luego multitud de cuestionarios temáticos que le sean aplicables.

PROGRAMA COMPUESTO CON SONIDOS, COLORES Y RESULTADOS

El programa no es del todo esquelético, pues viene compuesto con sonidos, colores y resultados. Además, una vez efectuada una sesión con él, se reinicializa sólo por si se quiere competir (científicamente) con algún compañero.

Veámoslo. En la línea 1, además de los pokes de vaciado del buffer (198) y de inutilización de la tecla RUN/STOP (788), hay una variable rara (M=RND(—TI)), que sirve para inicia-

(sigue el texto en la pág. 7)

APLICACIONES EDUCATIVAS



cómo hacer un cuestionario

por **J. ARGEMÍ**

LISTADO

```
READY.

1 CLR:M=RND(-TI):POKE198,0:POKE 788
,194

2 POKE 36879,25:POKE36878,15:S3=368
76:S2=36875

3 GOTO 27

4 RESTORE:FORJ=1TOINT(RND(TI)*30)+1
:READA$,B$:NEXT

5 POKE 788,194:V=V+1:IFV=11THEN44

6 PRINT"[ CLR ][CRSRD] SABES CUAL E
S LA CA-  PITAL DE :"

7 PRINT"[ GRN ] (ESCRIBELA Y PULSA
    LA TECLA RETURN).[ BLK ]"

8 PRINT"[CRSRD][ BLK ]"

9 PRINTTAB(2)A$:PRINT"[CRSRD][CRSRD]
"

10 PRINT" ";:POKE198,0:INPUTC$

11 D$=LEFT$(B$,3):E$=LEFT$(C$,3):IF
D$=E$THEN 17

12 PRINT"[CRSRD] LASTIMA !":PRINT"
DEBERIAS VIAJAR":PRINT" UN POCO MAS
  ..."

13 PRINT"[CRSRD][CRSRD] LA CIUDAD E
RA :":GOSUB68

14 PRINT"[ PUR ] [RVSON]";

15 PRINTB$"[ BLK ]"

16 FORJ=0TO8000:NEXT:GOTO 4

17 PRINT"[ CLR ][CRSRD][CRSRD] MUY
BIEN HOMBRE!      [CRSRD] MUY BIEN!"
:GOSUB 65:FORJ=0TO2000:NEXT

18 R=R+1:GOTO4

19 DATAFRANCIA,PARIS,INGLATERRA,LON
DRES,ITALIA,ROMA,PORTUGAL,LISBOA,SU
IZA,BERNA

20 DATAANDORRA,ANDORRA LA VELLA,MAL
TA,LA VALETA,GRECIA,ATENAS,POLONIA,
VARSOVIA

21 DATASUECIA,ESTOCOLMO,DINAMARCA,C
OPENHAGUE,NORUEGA,OSLO,ISLANDIA,REY
KIAVIK

22 DATALIECHSTENSTEIN,VADUZ,LUXEMBU
RGO,LUXEMBURGO,IRLANDA,DUBLIN,AUSTR
IA,VIENA

23 DATA YUGOSLAVIA,BELGRADO,BULGARI
A,SOFIA,HUNGRIA,BUDAPEST,RUMANIA,BU
CAREST

24 DATA ALEMANIA FEDERAL,BONN,ALEMA
NIA POPULAR,BERLIN,BELGICA,BRUSELAS

25 DATA UNION SOVIETICA,MOSCU,HOLAN
DA,AMSTERDAM,SAN MARINO,SAN MARINO

26 DATA FINLANDIA,HELSINKI,CHECOESL
OVAQUIA,PRAGA,TURQUIA,ANKARA

27 PRINT"[ CLR ]@@@@@@@@@@@@@@@@@@@
@@@";

28 FORJ=1TO19:PRINT"@
      @";:NEXT

29 PRINT"@@@@@@@@@@@@@@@@@@@@@@[HOME ]
"

30 PRINTTAB(3)"[CRSRU]GEOGRAFIA"

31 PRINTTAB(10)"[CRSRD]DE"

32 PRINTTAB(10)"[CRSRU] EUROPA"

33 PRINTSPC(248)

34 PRINT"[CRSRD][CRSRD][ GRN ]PULSA
 UNA TECLA[ BLK ]"

35 GET Q$:IFQ$=""THEN35

36 GOSUB 63:PRINT"[ CLR ][CRSRD][CRSRD]
 PARA FAMILIARIZARTE   CON LA GEOGR
AFIA DE   EUROPA ..."

37 PRINT"[CRSRD] TE HARE 10 PREGUNT
AS."

38 PRINT" SOBRE LAS CAPITALES   POL
ITICAS DE OTRAS    TANTAS NACIONES
EU-   ROPEAS."

39 PRINT"[CRSRD] DESPUES TE HARE SU
-   FRIR UN RATO, Y TE    DARE EL R
ESULTADO     FINAL."

40 PRINT"[CRSRD][ CYN ]******BUENA
SUERTE****[ BLK ][CRSRD]"

41 PRINTTAB(6)"[CRSRD][CRSRU][CRSRU]
[ GRN ]PULSA UNA TECLA[ BLK ]"

42 GETQ$:IFQ$=""THEN42

43 GOTO 4

44 IFR=0THENW$="SUSPENSO"

45 IFR=1THENW$="SUSPENSO"

46 IFR=2THENW$="SUSPENSO"

47 IFR=3THENW$="SUSPENSO"

48 IFR=4THENW$="SUSPENSO"

49 IFR=5THENW$="APROBADO"

50 IFR=6THENW$="APROBDO"

51 IFR=7THENW$="NOTABLE"

52 IFR=8THENW$="NOTABLE

53 IFR=9THENW$="SOBRESALIENTE"

54 IFR=10THENW$="MATRICULA[CRSRD][CRSRD]
[CRSRL][CRSRL][CRSRL]DE HONOR"

55 PRINT"[ CLR ]":FORJ=1TO2000:NEXT
:PRINT" [CRSRD][CRSRD][ RED ]TE EST
OY HACIENDO[CRSRD][CRSRD][CRSRL][CRSRL]
[CRSRL][CRSRL][CRSRL][CRSRL][CRSRL]
[CRSRL]EL DIPLOMA [ BLK ]"

56 FORL=250TO200STEP-2:POKES3,L:FOR
M=1TO100:NEXTM,L:POKES3,0

57 PRINT"[ CLR ]*******************
***";

58 FORJ=1TO19:PRINT"*
      *";:NEXT

59 PRINT"**********************[HOME ]
"

60 PRINTTAB(2)"[CRSRD][CRSRD][ GRN ]
EN EL DIA DE HOY"SPC(32)"MERECES"SP
C(32)"LA CALIFICACION DE :[ BLK ]"

61 PRINTTAB(6)"[CRSRD][CRSRD][CRSRD]
"W$:FORJ=0TO10000:NEXT

62 POKE198,0:GOTO1

63 POKES3,200:FORJ=1TO200:NEXT:POKE
 S3,0

64 RETURN

65 FORJ=50TO250:POKES3,J:NEXT:POKES
3,0:FORJ=150TO250:POKES3,J:NEXT:POK
ES3,0

66 FORJ=50TO250:POKES3,J:NEXT:POKES
3,0:FORJ=150TO250:POKES3,J:NEXT:POK
ES3,0

67 FORJ=50TO250:POKES3,J:POKES3,0:R
ETURN

68 POKES2,207:FORJ=1TO500:NEXT:POKE
S2,191:FORJ=1TO500:NEXT:POKES2,0

69 POKES2,207:FORJ=1TO500:NEXT:POKE
S2,191:FORJ=1TO500:NEXT:POKES2,0

70 RETURN

READY.
```

TRUCOS COMMODORE 64

cómo utilizar las teclas de función del COMMODORE 64

por **R.PARDO**

Las teclas de función en el COMMODORE 64 son un juego de cuatro teclas situadas en la parte derecha del teclado. Mientras usa un programa BASIC muchas veces es conveniente mostrar un mensaje como éste:

PARA CONTINUAR PULSE F1

Para el COMMODORE 64 es posible detectar estas teclas porque cada una tiene asignada su código CHR$. El margen de estos códigos es de 133 a 140 para las teclas F1 a F8. Una manera de detectar cuándo se pulsa la tecla F1 es la siguiente:

```
100 GET A$: IF A$<>CHR$(133)
THEN 100
```

Sin embargo, si no quiere usar CHR$ cada vez que quiera detectar una tecla de función, entonces tendrá que usar el método siguiente. Si escribe unas comillas (") y entonces pulsa una de las teclas de función, verá una serie de gráficos en inversión de video que aparecen en la pantalla. Puede utilizar esta característica de las teclas de función para hacer un «test» dentro de un bucle, y así facilitar la detección de la tecla:

```
100 GET A$: IF A$<> "[F1]"
THEN 100
```

Fíjese que "F1" entre corchetes significa que debe pulsar esta tecla, NO que lo escriba tal como se muestra; lo que verá entre comillas será un cuadrado sólido con una línea en la parte superior del cuadrado.

Otra manera de detectar las teclas de función (y cualquier otra tecla, de este modo) se realizará leyendo (PEEK) la posición de memoria 197 la cual tiene un código especial que le dice al ordenador qué tecla se ha pulsado. Esta posición contiene 64 cuando no se ha pulsado ninguna tecla y un número diferente si una tecla ha sido pulsada. Estos son los códigos que encontrará cuando una de las teclas de función haya sido pulsada:

F1 = 4
F2 = 5
F3 = 6
F4 = 3

Por ejemplo:

```
10 IF PEEK(197)<>4 THEN 10:
   REM ESPERA F1

20 IF PEEK(197)<>64 THEN 20:
   REM ESPERA LIBERACIÓN
   TECLA
```

Esperará que se pulse la tecla F1 y que se libere antes de seguir con el programa.

El siguiente programa BASIC le permitirá definir las teclas de función así como un mensaje de hasta ocho caracteres de longitud. Las líneas 200-330 entran la parte de código máquina a partir de la posición $C000. Si el programa se para en la línea 320 e imprime el mensaje de error, quiere decir que ha cometido un error a la hora de entrar las líneas DATA y deberá revisar esta parte del programa para corregirlas. Si los datos se introducen correctamente podrá hacer uso de las teclas de función y verá que a cada tecla se le ha asignado una palabra.

LISTADO

```
0 REM DEFINICION DE TECLAS DE FUNCI
ON POR R. PARDO
1 GOSUB300:REM ENTRA DATOS
2 F$(1)="LIST"+CHR$(13):REM PUEDE C
AMBIAR
3 F$(2)="RUN"+CHR$(13):REM LAS DEFI
NICIONES
4 F$(3)="GOSUB":REM RECORDANDO QUE
SON 8
5 F$(4)="SAVE"+CHR$(34):REM CARACTE
RES POR TECLA
6 F$(5)="LOAD":REM DE MAXIMO
7 F$(6)="GOTO"
8 F$(7)="CHR$("
9 F$(8)="RETURN"
10 SYS12*4096+64:REM INICIA CODIGO
20 V=12*4096-1:REM EMPIEZA DATO
30 FORI=1TO8:K=I-1:V1=V+K*8
40 FORJ=1TOLEN(F$(I)):REM ENTRA LAS
 DEFINICIONES DE TECLA
50 POKEV1+J,ASC(MID$(F$(I),J,1))
60 NEXT:NEXT:END
200 DATA120,169,87,141,20,3,169,192
,141,21
210 DATA3,88,162,63,169,0,157,0,192
,202
220 DATA16,250,96,165,197,201,64,20
8,6,141
230 DATA151,192,76,148,192,205,151,
192,240,44
240 DATA141,151,192,162,3,221,152,1
92,240,5
250 DATA202,16,248,48,29,138,174,14
1,2,240
260 DATA3,24,105,4,10,10,10,168,162
,0
270 DATA185,0,192,157,119,2,200,232
,224,8
280 DATA208,244,134,198,76,49,234,6
4,4,5,6,3,0
300 FORI=0TO92:READA:Z=Z+A
310 POKE12*4096+64+I,A:NEXT
320 IFZ<>10771THENPRINT"ERROR EN LA
S SENTENCIAS [RVSON]DATA[RVSOF]":ST
OP
330 RETURN
READY.
```

cómo hacer un cuestionario

(viene de la pág. 5)

lizar al azar, de forma diferente cada vez que carguéis el programa. En la línea 4, viene un valor muy importante para el funcionamiento del programa: es el *30 situado en el paréntesis del Random. Este valor es el número de posibles preguntas que hay en el cuestionario. Si alguna vez modificáis el programa o queréis aumentar el número de preguntas de éste, deberéis adecuar exactamente este valor al del número de preguntas que fijéis. En la línea número 5, existe otro valor importante: es la variable V (veces). Esta variable controla el número de preguntas que te hace el cuestionario en cada sesión. Si el valor *30 de la línea 4 da el total de preguntas sobre las que escoger, el valor V da en cambio el número de veces que serás preguntado por sesión. Como habréis agudamente observado, el programa elige aleatoriamente cada vez una pregunta entre las treinta posibles y todo esto lo hace sólo en una línea (la 4). En cambio os frustraréis si intentáis descubrir el porqué del valor 11 de la variable V, si sólo hace diez preguntas. Esto es debido a que, para ahorrar, hemos colocado al condicional (IF) en la misma línea, y si os fijáis en el contenido del condicional, veréis que dice que si V = 11 entonces no más preguntas. Así, si queréis cambiar el número de preguntas que el programa formule por sesión, cambiad simplemente el valor de la V contenida en el condicional (aumentando en uno el número que queráis). La línea número once compara tu respuesta con la verdadera y lleva a los mensajes de acierto (línea 17) o simplemente se deja caer sobre el desacierto (línea 12). La comparación se efectúa, para simplificar, sólo de las tres primeras letras de cada respuesta (aquí actúa el LEFT$).

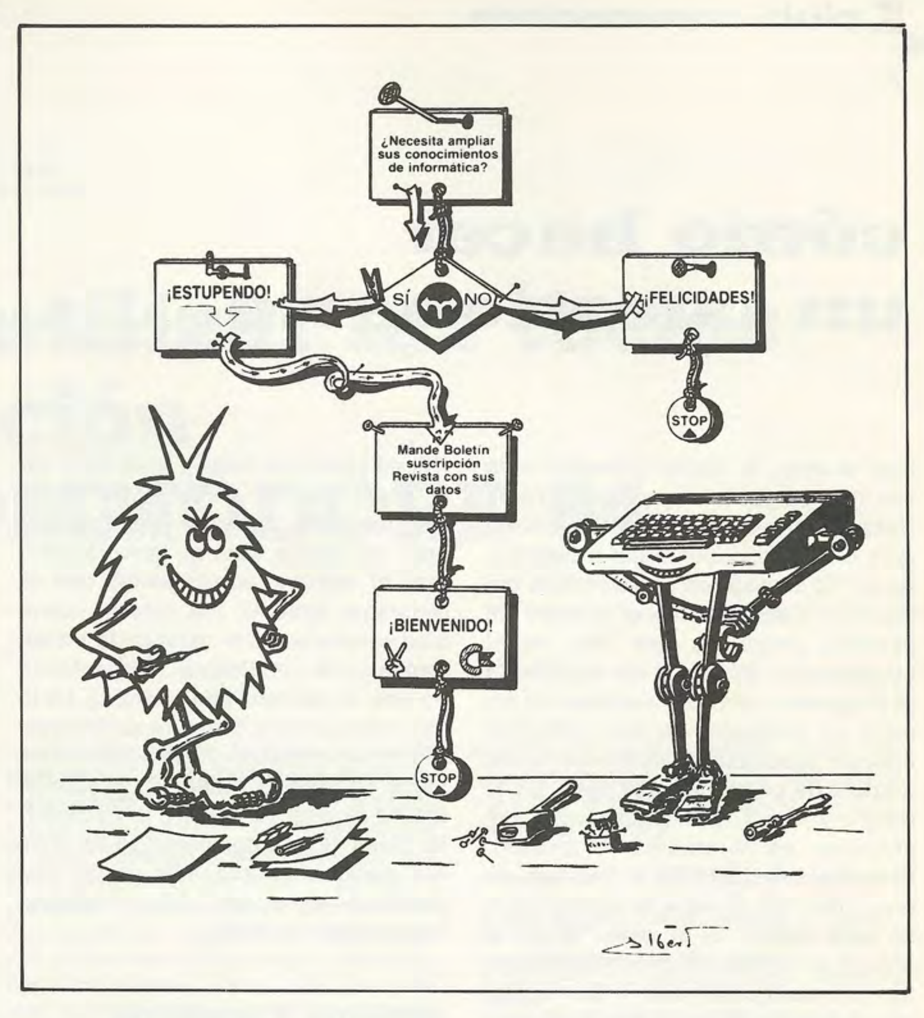
PREGUNTAS Y RESPUESTAS EN DATAS CONSECUTIVOS

No hace falta explicar el apartado de presentación (líneas 27 al 42) pero sí, en cambio, os habréis dado cuenta de que las preguntas y respuestas vienen dadas en DATAS consecutivos. Ingenioso ¿no? Estos DATA son además de fácil cambio y replanteamiento.

A partir de la línea 44 se dotan unas variables que proporcionan la información sobre la calificación que ha merecido una sesión del cuestionario. Esta tabla está confeccionada mediante la tradicional distribución académica, 0 al 4 Suspenso, 5 y 6 Aprobado, 7 y 8 Notable, 9 Sobresaliente y 10 Matrícula de Honor. A tener en cuenta que si queréis cambiar el número de preguntas por sesión (V), deberéis cambiar también los condicionales de las líneas 44 a la 54. O quizá también queráis cambiar el sistema de evalución... Con el Vic todo es posible.

El tema propuesto hoy de Geografía Política de Europa, es uno de los posibles que se adaptan bien a ser formulados en una sola pregunta: ¿Sabes la capital de...? Vosotros encontraréis muchos más (Elementos de Química, Valencias, etc.). Sólo deberéis incluir en los DATA a las preguntas seguidas de su correspondiente respuesta (no os olvidéis las comas) y la confección de estos cuestionarios os puede proporcionar una gran estratagema para pasar más ratos con el VIC, pues en definitiva estudiaréis Geografía o Química o... ■

clave para interpretar los listados de CLUB COMMODORE



Todos los listados que se publican en esta Revista han sido ejecutados en el modelo correspondiente de la gama de ordenadores COMMODORE. Para facilitar la edición de los mismos en la Revista y para mejorar su legibilidad por parte del usuario, se les ha sometido a ciertas modificaciones mediante un programa escrito especialmente para ello. Para los programas destinados a los ordenadores VIC-20 y COMMODORE 64, en los que se usan frecuentemente las posibilidades gráficas del teclado, se han sustituido los símbolos gráficos que aparecen normalmente en los listados por una serie de letras entre corchetes [ ] que indican la secuencia de teclas que se deben pulsar para obtener el carácter deseado. A continuación se da una tabla para aclarar la interpretación de las indicaciones entre corchetes:

[CRSRD] = Tecla cursor hacia abajo (sin SHIFT)
[CRSRU] = Tecla cursor hacia arriba (con SHIFT)
[CRSRR] = Tecla cursor a la derecha (sin SHIFT)
[CRSRL] = Tecla cursor a la izquierda (con SHIFT)
[HOME] = Tecla CLR/HOME (sin SHIFT)
[CLR] = Tecla CLR/HOME (con SHIFT)

Las indicaciones [BLK] a [YEL] corresponden a la pulsación de las teclas de 1 a 8 junto a la tecla CTRL. Lo mismo sucede con [RVSÓN] y [RVSOF] respecto a la tecla CTRL y las teclas 9 y 0.

El resto de las indicaciones constan de la parte COMM o SHIF seguidas de una letra, número o símbolo — por ejemplo [COMM+] o [SHIFA] —. Esto indica que para obtener el gráfico necesario en el programa deben pulsarse simultáneamente las teclas COMMODORE (la que lleva el logotipo) o una de SHIFT y la tecla indicada por la letra, el número o el símbolo, en el ejemplo anterior: COMMODORE y + o SHIFT y A, respectivamente. ■

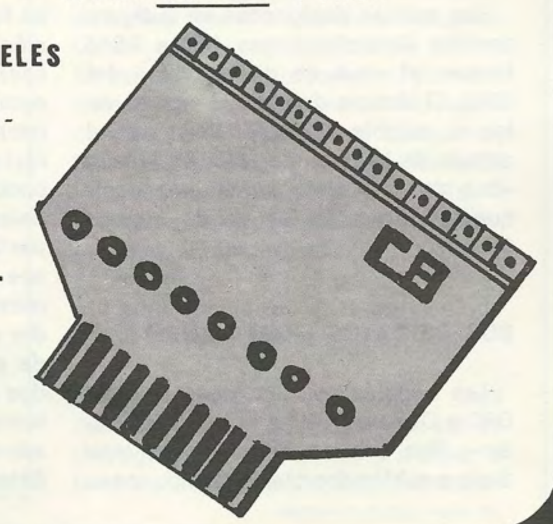
Elektrocomputer

ELEKTROCOMPUTER - PRESENTA SUS NUEVOS PRODUCTOS PARA EL VIC-20 Y EL COMMODORE-64. **DATAMASTER 64** Y **CONTROLADOR - C8**, QUE AMPLIAN LAS POSIBILIDADES DE SU ORDENADOR.

DE VENTA EN DISTRIBUIDORES AUTORIZADOS DE TODA ESPAÑA.

* **DATAMASTER** 64 _ SOFISTICADA BASE DE DATOS PARA EL C—64. PENSADA PARA TRABAJAR CON LA UNIDAD DE DISCO 1541. SIENDO MUY VERSATIL APROVECHA AL MAXIMO LA CAPACIDAD DE MANIOBRA Y ALMACENAMIENTO. NUMERO DE REGISTROS VARIABLE "EJ. 5000 DE 30 CARACTERES". FORMATEADOS Y COPIAS PROGRAMADAS. SALIDA A IMPRESORA (PARALELO CENTRONICS Y SERIE RS 232) CHEQUEO OPERACIONES DISCO. GARANTIA 3 MESES. MANUAL COMPL. EN CASTELLANO — P.V.P. 11.800' PTAS.

* **CONTROLADOR - C8** _ CONTROLADOR DE 8 RELES PARA EL VIC-20 Y EL C-64. DE FORMA MUY SENCILLA PODEMOS HACER HASTA 255 COMBINACIONES ENTRE LOS 8 RELES, CON UN CONSUMO DE 1000 W. A 220 VOLT. CADA UNO. CON LO CUAL PODEMOS ACCIONAR TODO TIPO DE LUCES O MECANISMOS. INSTRUC. INCLUIDAS. 3 MESES GARANTIA — P.V.P. 9.800' PTAS.



VIA AUGUSTA - 120 - TEF. (93 - 2180699 - BARCELONA - 6

código máquina del 6502 (III)

por **P. MASATS**



EL 6502 PATILLA A PATILLA

Vamos a ver ahora las funciones de cada una de las conexiones de nuestro circuito integrado. Para ello vamos a seguir un orden más o menos lógico en vez de seguir el orden numérico:

ALIMENTACIÓN

Vss y Vcc, patillas 1, 8 y 21. En estos tres puntos se aplica una tensión continua de 5 V ± 5 % que alimentará a nuestro microprocesador. En Vcc (8) se aplica el positivo y en los dos Vss (1 y 21) el negativo o masa.

BUS DE DIRECCIONES («ADDRES BUS»)

Las patillas designadas en la figura con las denominaciones AB0 a AB15 forman el «bus de direcciones» del 6502. El tiempo durante el cual el valor es estable viene señalado por el estado de la línea de reloj 01 (véase «bus de control»), pudiéndose decir que la dirección es válida cuando 01 = 1.

BUS DE DATOS («DATA BUS»)

Las señales con las designaciones DB0 a DB7 constituyen el «bus de datos». Sus valores deben haber quedado establecidos cierto tiempo antes de que 02 (véase «bus de control») pase de 1 a 0, tanto en lectura como en escritura. Debemos hacer una advertencia sobre los detalles de tiempos de validez de datos y direcciones: las explicaciones que hemos dado son algo esquemáticas pero cumplen la función de informar al lector de lo que debe saber sin sumergirlo en procelosos mares de información técnica.

«BUS DE CONTROL»

00, 01 y 02 (pronúnciese «fase» cero, uno y dos), patillas 37, 3 y 39. Un microprocesador es una máquina de tipo secuencial, lo que quiere decir que realiza su trabajo (ejecutar las instrucciones del programa) una detrás de otra con una cadencia fija. Pues bien, la velocidad de esta cadencia viene determinada por una señal en forma de onda cuadrada (una sucesión alternativa de ceros y unos lógicos) que generada por un dispositivo exterior se aplica al terminal 00. El microprocesador, por su parte suministra al resto del sistema (memorias, controladores de video, etc...) dos señales del mismo tipo que indican a los periféricos —no debe olvidarse que nos referimos a los periféricos del microprocesador, no a los del ordenador— en qué parte de la secuencia de ejecución está el procesador. Las dos señales 01 y 02 suministran al sistema esta información. Como debe adivinarse, la frecuencia de esta señal determina la velocidad con la que el microprocesador realiza sus operaciones. En el caso del 6502 esta velocidad suele ser de un millón de operaciones por segundo con una frecuencia de 1 MHz. Por sus características de control temporal a este conjunto de señales se las conoce normalmente como **«reloj del sistema».**

RDY patilla 2. — Si se lleva a un valor 0 a esta línea se fuerza una detención momentánea en la ejecución de instrucciones por parte del microprocesador. Su principal utilidad es la de permitir el trabajo con memorias lentas (que tardan en suministrar la información que contienen una vez se les ha dado la dirección deseada) sin necesidad de alterar la velocidad general del sistema.

IRQ, patilla 4. — Su nombre es «demanda de interrupción enmascarable» (Interrupt Request). Para analizar su función empecemos por la parte «demanda de interrupción». Supongamos que estamos realizando una tarea que requiera toda nuestra atención, programando un ordenador —por ejemplo— y, al mismo tiempo, esperamos una llamada importante por teléfono. ¿Cómo podemos solucionar el problema de compaginar las dos actividades? La solución inmediata consiste en dejar de programar a intervalos regulares y verificar si el teléfono está sonando. En caso afirmativo, atendemos la llamada; en caso negativo, proseguimos nuestro trabajo donde lo habíamos dejado. Esta solución tiene

varios inconvenientes: debemos emplear una cantidad de tiempo a intervalos fijos para interrogar el teléfono tanto si existe una llamada como si no y si ésta se produce en el momento justo en que hemos dejado de prestar atención al aparato no lo atenderemos hasta que empecemos otro ciclo de interrogación. La mejor solución es la que utilizamos normalmente: realizar nuestra actividad principal hasta que el timbre del teléfono **nos interrumpa.** Así conseguimos dedicarnos a nuestra actividad principal a tiempo completo y dedicar al teléfono solamente el tiempo necesario. La señal IRQ que en la figura aparece con una raya encima actúa de la misma manera. Esta línea cubriendo el nombre de una señal indica que ésta es activa cuando pasa a valor 0 (al contrario de lo que es normal en lógica digital); luego, cuando la señal IRQ pase de 1 a 0 se interrumpirá la tarea que esté realizando el microprocesador y se empezará a ejecutar una parte del programa a partir de una posición de memoria determinada, en el caso de que esta interrupción no esté «enmascarada». Mediante el valor de un bit, en una posición concreta de un registro interno del 6502, podemos evitar que IRQ interrumpa al procesador, con lo que podemos conseguir lo mismo que desconectando el teléfono de nuestro ejemplo: la llamada puede producirse pero no tendremos ninguna interrupción. A esta posibilidad se le llama «enmascarar» la interrupción.

NMI, patilla 6. — El nombre de esta señal es «interrupción no enmascarable» («Non-Maskable Interrupt»). Su función es la misma que IRQ excepto que no podemos evitar que actúe (no es enmascarable). Por llevar la raya encima sabemos que actúa en el paso de 1 a 0.

Vss	1	40	$\overline{RES}$
RDY	2	39	Ø2 (OUT)
Ø1 (OUT)	3	38	S.O.
$\overline{IRQ}$	4	37	Ø0 (IN)
N.C.	5	36	N.C.
$\overline{NMI}$	6	35	N.C.
SYNC	7	34	R/W
Vcc	8	33	DBØ
ABØ	9	32	DB1
AB1	10	31	DB2
AB2	11	30	DB3
AB3	12	29	DB4
AB4	13	28	DB5
AB5	14	27	DB6
AB6	15	26	DB7
AB7	16	25	AB15
AB8	17	24	AB14
AB9	18	23	AB13
AB10	19	22	AB12
AB11	20	21	Vss

MCS6502

fig. 1 - NC = NO CONECTADO

RES, patilla 40. — Éste es un tercer tipo de interrupción aunque normalmente no se la considera como tal. Su nombre es «inicialización» (RESet) y su función principal es la de suministrar al microprocesador una señal que le indique cuándo se conecta la tensión para realizar las tareas de preparación del equipo para su funcionamiento, dado que, por ejemplo, las memorias RAM tienen valores aleatorios cuando se les da corriente después de habérsela cortado. En los tres casos de interrupción existen dos posiciones de memoria reservadas para almacenar el valor de una dirección a partir de la cual empezará a ejecutarse un programa para cada una de las tres. Estas direcciones son:

IRQ - $FFFE y $FFFF
NMI - $FFFA y $FFFB
RES - $FFFC y $FFFD

A este tipo de almacenamiento se le llama almacenamiento de punteros y, como puede verse, éstos ocupan los seis bytes más altos del mapa de memoria.

SYNC, patilla 7. — En el 6502 el programa a ejecutar por la CPU está almacenado en memoria, el microprocesador toma de ésta el código que debe ejecutar, averigua de qué instrucción se trata y procede a su ejecución, para volver a empezar el ciclo de nuevo. La señal SYNC (sincronismo) permite detectar en qué momento el microprocesador está leyendo un código de instrucción, con lo que resulta fácil — en conjunción con la línea RDY — en ciertas aplicaciones que el microprocesador ejecuta las instrucciones de una en una, lo cual puede ser muy útil para encontrar fallos en la etapa de diseño de sistemas.

S. O., patilla 38. — Mediante esta conexión se puede obligar a un bit de un registro interno del 6502 a pasar a tener un valor de 1. Este bit es el

(termina en la pág. siguiente)

código máquina del 6502

(viene de la pág. anterior)

que también detecta el rebosamiento («overflow») en las operaciones aritméticas. Su uso es muy limitado.

R/W, patilla 34. — Ésta es una señal de gran importancia debido a que indica a los periféricos del 6502 si éste realiza una operación de lectura (READ) o escritura (WRITE) en memoria. Para realizar, por ejemplo, una operación de suma de dos números almacenados en memoria, hay que tomar el primero de ellos, almacenarlo en el registro aritmético del microprocesador; tomar el segundo, sumarlo al acumulador (que es el nombre con el que se suele designar al registro A o aritmético) y almacenar el resultado en una tercera posición de memoria para su tratamiento posterior. Como es lógico necesitamos una señal que indique a la memoria si debe poner el valor almacenado en la dirección de memoria indicada por el BD* en el BN* (lectura) o debe memorizar el valor del BN* en la celda de memoria existente en el BD* (escritura). Si R/W es igual a 1 se realizará una operación de lectura y si es 0 de lectura.

* * *

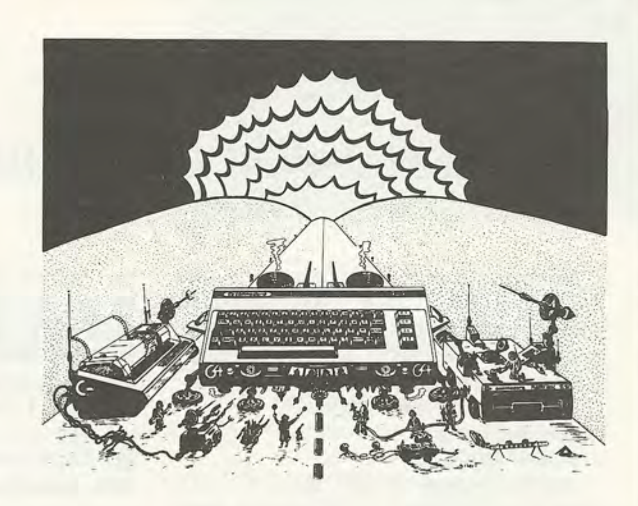
Con lo que antecede hemos dado un somero repaso al «patillaje» del 6502. Solamente queda por decir que las patillas marcadas en la figura con las letras N. C. no realizan ninguna función (No Conectadas).

Quizás alguien encuentre algo superficial esta explicación pero esta serie de artículos está dirigida a los aficionados a la microinformática para que puedan iniciarse en el microprocesador sin desilusiones. En todo caso, pueden profundizar en sus conocimientos posteriormente recurriendo a obras de mayor profundidad una vez comprendida esta introducción. ■

AVISO

El exceso de originales para publicar y la falta de espacio para incluirlos en su totalidad, nos obliga a dejar para próximas ediciones algunos de los capítulos de las series que no quedan suspendidas sino tan sólo retenidas.

COMMODORE 64



arquitectura del COMMODORE 64 (y II)

por **R. PARDO**

En la última edición de nuestra Revista vimos que, al disponer el COMMODORE 64 de ¡8 mapas de memoria distintos!, gozamos de una flexibilidad increíble para sustituir las secciones ROM de BASIC y de KERNAL por ROMs exteriores (en forma de cartucho) o por un lenguaje que se cargue desde disco. Y todo esto sin perder memoria RAM. Proseguimos ahora enumerando las ventajas que proporciona el 64 a sus usuarios.

UNA DE LAS CARACTERÍSTICAS MÁS SOBRESALIENTES DEL COMMODORE 64

Esperando que hayáis comprendido la estructura básica del mapa de memoria de la máquina, veremos otra de las características más sobresalientes de la máquina, y, cosa curiosa, no se comenta en ninguno de los folletos de propaganda del COMMODORE 64. En un ordenador clásico como el VIC-20 o la serie 4000 u 8000 de COMMODORE, sabemos que si hacemos POKE a una posición de memoria ocupada por una ROM, el valor POKEado a esta posición no se guardará en ningún sitio y que cuando hagamos PEEK de esa misma posición nos devolverá el valor grabado en esta posición de la ROM. Pero el COMMODORE 64 es un ordenador vanguardista. Cuando hacemos POKE a una posición de memoria donde el procesador «ve» ROM, el valor que queremos guardar se almacena en la misma posición, pero en la RAM que está debajo. Sin embargo, si hacemos PEEK de esta misma posición nos devolverá el valor grabado en la ROM.

Aprovechando esta característica tan apasionante, podríamos hacer ... mmm ... podríamos copiar el BASIC o el KERNAL en RAM y modificarlos de alguna manera...

Veamos cómo podríamos hacer esto. El primer paso consistiría en copiar la ROM en la RAM que hay debajo. El ejemplo que voy a dar es el de la copia de la ROM de BASIC. Entre el siguiente programa y haga RUN.

```
100 FOR J = 40960 TO 49151
110 POKE J, PEEK (J)
120 NEXT J
```

Tardará aproximadamente un minuto en ejecutar este programa. Una vez aparezca el mensaje READY. en la pantalla, tendremos ya una copia exacta de la ROM de BASIC en RAM. Ahora hay que decirle al procesador que debe utilizar la copia que hemos hecho. Vea la tabla anterior de POKEs y escriba:

```
POKE 1, 54
```

El ordenador le contestará READY. y el cursor aparecerá debajo de este mensaje como si nada hubiera pasado, pero en realidad ¡¡el ordenador está usando ya la copia de BASIC que hemos creado anteriormente!!

Para probar que realmente está en RAM, vamos a hacer unos cambios en el BASIC.

FORMA DE SUBSANAR UN ERROR

Hay una situación muy corriente de error que se da trabajando con el BASIC standard que viene en todas las máquinas de COMMODORE y es que si se hace un PRINT ASC (" ") donde la cadena entrecomillada está vacía le dará un error del tipo ?ILLEGAL QUANTITY ERROR.

(termina en la pág. siguiente)

arquitectura del COMMODORE 64

(viene de la pág. anterior)

Nosotros como disponemos de una copia del BASIC en RAM subsanaremos este inconveniente, es decir, cuando se encuentre ante esta condición que devuelva el valor 0, que sería para mí lo más lógico.

Escriba pues:

POKE 46991, 5

Ahora pruebe PRINT ASC ("") y verá cómo le da 0. ¡¡Ya hemos cambiado el BASIC!! Pero aún podemos hacer más. Podemos, por ejemplo, cambiar la sintaxis de una palabra BASIC de la misma manera. LISTe el programa anterior (el que hacía la copia) y escriba:

POKE 41122, 69

y vuelva a listar el programa. Notará un cambio en la línea 100 del programa:

```
100 FER J = 40960 TO 49151
110 POKE J, PEEK(J)
120 NEXT J
```

¿Ve qué fácil? Pero puede ser más divertido aún. En un ataque de distracción escriba:

POKE 41230, 85

Y ahora LISTe el programa. Probablemente tropezará con un pequeño inconveniente... no podrá listar el programa porque... ¡¡LIST ya no es LIST!! Pero no se preocupe, para listar tendrá que escribir LUST. Como se puede ver, es realmente sencillo modificar el BASIC o el KERNAL a nuestro antojo. De todos modos, si quiere volver a la normalidad escriba:

POKE 1, 55

EL PROCESADOR SE ENCARGA DEL MANEJO DEL CASSETTE

Antes hemos hablado también del port de Entrada/Salida que lleva integrado el microprocesador. Para quien conozca un poco esta clase de integrados le parecerá un poco extraño el hecho de que sólo se utilicen los tres primeros bits del port de E/S del 6510. En realidad, se usan los seis primeros bits del port.

Los otros tres bits tienen que ver con el manejo del port de cassette. ¡¡Sí, sí, el propio procesador se encarga del manejo del cassette!!

Éstas son las funciones que ejecutan cada una de estas tres líneas (cada bit = una línea):

1. El bit 3 controla la línea de escritura del cassette y esta línea está configurada como salida.
2. El bit 4 controla los sensores de las teclas del cassette. Esta línea está configurada como entrada.
3. El bit 5 controla el motor del cassette, siendo ésta una línea de salida.

Ésta es la razón de que cuando se opera con el cassette (operaciones de lectura/escritura) se «vaya» la pantalla, ya que el microprocesador debe encargarse personalmente de la utilización del cassette.

Otra característica especial del COMMODORE 64 es el «blanking» (borrado) de pantalla en las operaciones con el cassette y con la unidad de discos 1540.

Los usuarios que dispongan de esta unidad de discos ya saben que deben hacer, ANTES de cualquier operación de disco, un POKE al controlador de video para «desactivar la pantalla»:

POKE 53265, 11

y para volverla a activar:

POKE 53265,27

UNA FORMA DE INCREMENTAR LA VELOCIDAD DEL SISTEMA

Ahora bien, este bit que ponemos a 1 ó a 0, según nos convenga, ha sido creado con una finalidad específica: liberar el bus del procesador.

Como ya se comentó en el número 7 de esta Revista, el control del bus del microprocesador lo ejercían por igual la CPU (el procesador central) y el chip de video. ¿Por qué? Muy sencillo. Los 64 K de RAM de los que dispone el ordenador son RAMs dinámicas 4164 (65536 × 1 bits). Esto quiere decir que se necesita un chip que se dedique a refrescar la información que contienen estas memorias (qué es y cómo funciona una RAM dinámica es algo que excede los límites de este artículo y, por lo tanto, no se explicará). El chip que se encarga de esto es el propio chip de video. Además, este chip «maneja» 16 K de RAM a la vez por motivos que se pueden ver a simple vista: tener acceso a memoria de pantalla, memoria de color, patrones de «sprites», generador de caracteres y pantalla de alta resolución.

Este bit lo que nos permite es, como hemos dicho antes, liberar el bus del procesador. ¿Qué ganamos con esto? Incrementar la velocidad del sistema. En aplicaciones donde se programe en ASSEMBLER y la velocidad sea un factor crítico y además no se necesite una presentación constante de datos por pantalla, es recomendable ejecutar esta operación.

Y con esto creo que muchas preguntas que se nos hacían han hallado una respuesta adecuada. ■

SOFTWARE DE BASE (VIII)

la rutina de Pasthel

por **E. MARTÍNEZ DE CARVAJAL**

Con este artículo abro un paréntesis en esta sección, SOFTWARE DE BASE, para iniciar otra serie. Para esta ocasión hemos seleccionado una rutina que con seguridad os dejará pasmados y que, con el permiso de la concurrencia, quisiera dedicar a las siguientes personas: Margarita, Mariana, Vicente, Fernando, Carlos, Ana, Bibiana, Alex, Liberto, Coloma, Sonia, Jaime, Quim, Mariángeles, Juan, José, Carmelita, Maricarmen, Maribel, Paz, Tania, Pere, Pilar, M. Mar, Félix, Francisco, Pedro, Javier, Jaume, a todos los lectores de CLUB COMMODORE y a .... (aquí ponga su nombre).

Ésta es una rutina que permite calcular de forma rápida y muy exacta la media aritmética de cinco números. Fue desarrollada por el filósofo y matemático Pasthel en el año 1920 y modificada posteriormente por su discípulo Pierre Masanet en el año 1944 adaptándolo a los ordenadores de su época.

No os detallo su funcionamiento puesto que en la rutina he incluido los comentarios originales traducidos al castellano que explican claramente su funcionamiento paso a paso.

Se puede modificar fácilmente para que sirva para cualquier cantidad de números. Para ello hay que añadir los datas correspondientes en la línea 120, modificar la línea 200 adaptándola al nuevo número de datas, y calcular el nuevo factor de corrección según la fórmula:

Valor = Sumatorio(1/(n-1)) con n entre 1 y el número de valores y ponerlo en lugar del valor 3.1415926.

Como podéis ver el programa es relativamente corto teniendo en cuenta que en el resultado el error máximo es ± 0.00001. ■

BARCELONA, 28 DICIEMBRE 1983

LISTADO

```
READY.


10 REM PROGRAMA PARA CALCULAR LA MEDIA D
E CINCO NUMEROS
20 REM
30 REM METODO DE "PASTHEL ORGANIZATION"
40 REM
100 REM ------------------------------
110 REM DATAS DE LOS 5 NUMEROS
120 DATA 2,4,8,7,6
130 REM ------------------------------
180 REM SUMA DE LOS NUMEROS
190 T=0
200 FOR I=1 TO 5
210 READ N
220 T=T+N
230 NEXT I
240 T1=T
260 REM CALCULO DE LA DESVIACION MEDIA O
RTOGONAL
270 M=(T*3.1415926)/5-(T+1)
280 M=M*5/2
290 REM ------------------------------
300 REM CALCULO DE LAS VARIACIONES MAXIM
AS BILATERALES
310 V1=INT(T-M)/2
320 V2=INT(T+M)/2
390 REM ------------------------------
400 REM INTERPOLACION DE VALORES Y FILTR
AJE DE MAXIMOS
410 V3=V1+V2/2
420 T=T-(V3/2)
490 REM ------------------------------
500 REM APLICACION DEL PASTHEL
510 T=M*5+(V1+V2)/2-V3
520 T=T1/5
530 T1=M*5-(V1+V2)/2+V3
590 REM ------------------------------
600 REM IMPRESION DEL RESULTADO
610 PRINT "MEDIA APROXIMADA (E=+-0.00001
) : ";T
690 REM ------------------------------

READY.
```

MARKETCLUB

● CBM 4.032. Intercambio programas. José Marcé Mestres. Calle Sevilla, 5. Tel. (93) 803 77 51, de 8 a 3. VILANOVA DEL CAMÍ.

● ACCESORIOS VIC-20: Ampliación de memoria 16K más varios programas, 13.000. Módulo de expansión para 6 cartuchos, 10.000. Cartucho lenguaje FORTH y manual, 7.000. Las tres cosas sólo por 25.000. Jaime. Tel. 245 46 56. BARCELONA.

● En Barcelona, clases de informática. PLAZAS LIMITADAS. Lenguaje BASIC. Prácticas con micro-ordenador VIC-20. Prof. E. Martínez de Carvajal. Información: Tel. (93) 345 10 00. Señorita María José (mañanas) o (93) 345 87 75. Sr. Martínez (fuera de horas de oficina).

● Desearía vender por 40.000 ptas.: VIC-20, con cartucho Super Expander, las dos partes del Curso, el «joy-stick» y un juego Indescomp, todo comprado en diciembre del 82. Fernando Martínez, calle La Roda, 39, 5º D. Teléfono 23 41 82. ALBACETE.

● Tengo un PET 2001/8K y desearía tener el cassette «Monitor para lenguaje Máquina». También desearía contactar con usuarios o clubs del PET si los hay. José Manuel Cámara Mas, calle Castor, 32, bloque II, 3º, puerta 1. ALICANTE.

● Programación de ordenadores personales; organización explotación de ficheros; programas ordenadores auxiliares, para cuestiones empresariales, profesionales, administrativas, científicas. Mora Mas. Carlos III, 41. Teléfono 339 98 29. BARCELONA-28.

● Vendo aplicación de facturación con control de representantes, 9 listados, 6 ficheros, estadísticas, etc. Permite copias de seguridad. Configuración: VIC 8 K, disco e impresora, 40.000 pesetas. Escribir a Jaime Ameller Pons. General Mola, 15, 1º B. CALATAYUD (Zaragoza).

● Se ofrece 3008 + C2N de segunda mano en buenas condiciones; precio a convenir. Razón: Domingo Garrofé Trabal. Aragón, 386, 1º 1ª Tel. 211 54 40. BARCELONA-9.

● Vendo interfaz y programa para RTTY y CW para el PET a 15 K. Rafael, EA3CGK. Av. Barcelona, 21, A, 4º 2ª. IGUALADA (Barcelona).

EJEMPLARES ATRASADOS DE «CLUB COMMODORE»

Para poder satisfacer la creciente demanda de números atrasados de nuestra Revista, agotada en todas sus ediciones, hemos puesto en marcha un Servicio para suministrar fotocopias de los ejemplares que nos sean solicitados. Para recibir las fotocopias de una o de varias ediciones, no hay más que enviarnos el boletín con los datos indicados.

SERVICIO DE FOTOCOPIAS

NÚMERO DE LA EDICIÓN SOLICITADA

1	2	3	4	5	6	7	8	9	10	11	12

(Poner una X debajo del número de edición pedido)

Peticionario: D. ....

Calle .... n.° ....

Población .... D.P. ....

Provincia ....

Forma de envío: contra-reembolso **Precio de la edición fotocopiada: 250 ptas.**

MAPAS DE MEMORIA DEL COMMODORE 64

Address	Memory
	8K KERNAL ROM
E000	4K I/O
D000	4K RAM
C000	16K RAM
8000	16K RAM
4000	16K RAM
0000	

X = DON'T CARE
0 = LOW
1 = HIGH

LORAM = 0
HIRAM = 1
GAME = 1
EXROM = X

This map is intended for use with softload languages (including CP/M), providing 52K contiguous bytes of user RAM, I/O devices, and I/O driver routines.

Address	Memory
	8K KERNAL ROM
E000	4K I/O
D000	4K RAM (BUFFER)
C000	8K ROM (CARTRIDGE)
A000	8K RAM
8000	16K RAM
4000	16K RAM
0000	

X = DON'T CARE
0 = LOW
1 = HIGH

LORAM = 0
HIRAM = 1
GAME = 0
EXROM = 0

This map provides 40K contiguous bytes of user RAM and up to 8K bytes of plug-in ROM for special ROM-based applications which don't require BASIC.

Address	Memory
	16K RAM
C000	16K RAM
8000	16K RAM
4000	16K RAM
0000	

X = DON'T CARE
0 = LOW
1 = HIGH

LORAM = 0
HIRAM = 0
GAME = 1
EXROM = X
OR
LORAM = 0
HIRAM = 0
GAME = X
EXROM = 0

This map gives access to all 64K bytes of RAM. The I/O devices must be banked back into the processor's address space for any I/O operation.

Address	Memory
	8K KERNAL ROM
E000	4K I/O
D000	4K RAM (BUFFER)
C000	16K ROM (CARTRIDGE)
8000	16K RAM
4000	16K RAM
0000	

X = DON'T CARE
0 = LOW
1 = HIGH

LORAM = 1
HIRAM = 1
GAME = 0
EXROM = 0

This map provides 32K contiguous bytes of user RAM and up to 16K bytes of plug-in ROM for special ROM-based applications which don't require BASIC (word processors, other languages, etc.).

Address	Memory
	8K KERNAL ROM
E000	4K I/O
D000	4K RAM (BUFFER)
C000	8K BASIC ROM
A000	8K ROM CARTRIDGE (BASIC EXP)
8000	16K RAM
4000	16K RAM
0000	

X = DON'T CARE
0 = LOW
1 = HIGH

LORAM = 1
HIRAM = 1
GAME = 0
EXROM = 0

This is the standard configuration for a BASIC system with a BASIC expansion ROM. This map provides 32K contiguous bytes of user RAM and up to 8K bytes of BASIC "enhancement."

Address	Memory
	8K CARTRIDGE ROM
E000	4K I/O
D000	4K OPEN
C000	8K OPEN
A000	8K CARTRIDGE ROM
8000	16K OPEN
4000	12K OPEN
1000	4K RAM
0000	

X = DON'T CARE
0 = LOW
1 = HIGH

LORAM = X
HIRAM = X
GAME = 0
EXROM = 1

This is the ULTIMAX video game memory map. Note that the 2K byte "expansion RAM" for the ULTIMAX, if required, is accessed out of the COMMODORE 64 and any RAM in the cartridge is ignored.



Microprocesador: 6502 de MOS TECHNOLOGY de 8 bits.
Memoria: 5 Kbytes de RAM ampliables a 32 K 20 Kbytes de ROM ampliables a 28 K.
Pantalla: 23 líneas de 22 caracteres. Modulador para conectar a un televisor normal. Salida monitor video.
Colores: 8 para el marco, 16 para el fonde de la pantalla y 8 para los caracteres individuales, video inverso.
Gráficos: Semi-gráficos por teclado y alta resolución por redefinición del generador de caracteres (situándolo en RAM). Definición de 176 por 184 puntos.
Teclado: Tipo QWERTY de 62 teclas más cuatro de función definibles por el usuario.
Sonido: Tres voces de tres octavas cada una decaladas una octava entre si, resultando una extensión total de cinco octavas. Un generador de ruido aleatorio afinable para efectos especiales, un control general de volumen.
Programación: Lenguaje BASIC, intérprete residente en ROM de 8K. Posibilidad de interceptar las funciones del Basic para crear nuevas instrucciones "a medida". El Basic del Vic es uno de los rápidos actualmente en el mercado.
Complementos: Port de usuario de 8 bits entrada/salida más dos señales de sincronismo.
Bus de expansión para ampliaciones de memoria y periféricos.
Port de juegos con conexión para dos potenciómetros (paddles), y una palanca de juegos (joystick).
Almacenamiento de masa: Unidad de cassette C2N de diseño especial para registrar programas y datos.
Ampliación de memoria: En caso de ser necesario conectar más de un cartucho al mismo tiempo, está disponible un módulo (VIC 1020) que permite la conexión simultánea de hasta seis cartuchos.

VIC-1541 UNIDAD DE DISCO

Capacidad total: 174848 bytes por disco.
Secuencial: 168656 bytes por disco.
Entradas de directorio: 144 por disco.
Sectores por pista: De 17 a 21.
Bytes por sector: 256.
Pistas: 35.
Bloques: 683 (644 bloques libres).
Soportes de información: Discos standar de 5 1/4 pulgadas, de una sola cara y densidad simple.
Sistema operativo: DOS de COMMODORE inteligente (tiene procesador propio y no ocupa memoria del ordenador central).

VIC-1525 IMPRESORA

Método de impresión: Matriz de 5 × 7 puntos, impacto por un solo martillo.
Modo caracteres: Mayúsculas y minúsculas, símbolos, números y caracteres gráficos del VIC-20.
Modo gráfico: Puntos direccionables (bit image). Siete puntos verticales por columna, 480 columna máximo.
Velocidad: 30 caracteres/segundo, de izquierda a derecha, unidireccional.
Caracteres/Línea: Máximo 80. (Posibilidad de impresión en doble ancho).
Espaciado entre líneas: 6 líneas/pulgada —modo caracteres, 9 líneas/pulgadas— modo gráfico.
Alimentación de papel: Arrastre por tractor.
Ancho de papel: Entre 4,5 y 10 pulgadas.
Copias: Original más dos copias.

CARTUCHOS

Ayuda programador: Facilita la edición y depuración de programas en Basic. Instrucciones y comandos: RENUMBER, MERGE, FIND, CHANGE, DELETE, AUTO, TRACE, STEP, OFF, KEY, EDIT, PROG, DUMP, HELP y KILL.

Super expander: Intercepta el Basic del VIC permitiendo incrementar sus instrucciones y comandos en aplicaciones gráficas de sonido y juegos. Instrucciones y comandos: KEY, GRAPHIC, COLOR, POINT, REGION, DRAW, CIRCLE, PAINT, CHAR, SCNCLR, SOUND, RGE, RCOLR, RDOT, RPOT, RPEN, RJOY y RSND.
Monitor de lenguaje máquina: Facilita enormemente la depuración de programas en lenguaje máquina, es ideal como complemento del Basic para redactar y poner en marcha rutinas de alta velocidad y manejo de datos en tiempo real. Instrucciones y comandos: ASSEMBLE, BREAKPOINT, DISASSEMBLE, ENABLE, VIRTUAL ZERO PAGE, FILL MEMORY, GO, HUNT, INTERPRET, JUMP TO SUBROUTINE, LOAD, MEMORY, NUMBER, QUICK TRACE, REGISTERS, REMOVE BREACPOINTS, SAVE, TRANSFER, WALK y EXIT TO BASIC.
Además existen cartuchos de ampliación de memoria de 3,8 y 16 Kbytes.

CURSO DE INTRODUCCION AL BASIC PARTE I y II.

En forma de libro se ha editado la primera y segunda parte de un curso de Basic que parte "de cero" y está basado en el VIC-20. Van acompañados de dos cassettes con programas y ejercicios para autocontrol.

PLOTTER VIC 1520

Método de impresión: Dibujo mediante bolígrafos de diseño especial.
Color: 4 colores; negro, azul, verde y rojo con cambio desde programa.
Cabezal: Ploter X-Y tipo tambor.
Velocidad de impresión: Media de 14 car./seg.
Caracteres por línea: Máximo 80 carac., formatos de 80, 40, 20 y 10 carac./línea.
Juego de caracteres: 96.
Velocidad de dibujo: 264 pasos/seg.
Longitud del paso: 0,2 mm. en dirección X e Y.
Velocidad de dibujo de línea: 52,8 mm./seg. en dirección X e Y. 73 mm./seg. en una línea a 45 grados.
Area de dibujo: 480 pasos (96 mm.) en dirección X. Programable en dirección Y (Máx. + — 999 de una sola vez).
Papel: Rollo de 4,5 pulgadas (114 mm.).

MONITOR EN COLOR C-1701

Pantalla: 13 pulgadas (330 mm.).
Capacidad de representación: 25 líneas de 40 caracteres.
Resolución: 320 líneas horizontales.
Compatibilidad: VIC-20 y COMMODORE 64.
Conectable a un registrador de video.
Amplificador y altavoz: Incorporados.

